**NLP Test notes**

Java and maven version

I was able to build and run the project on Java 7 and Java 8 but I wrote the code on a machine running Java 8 and maven 3.2.5 in case you had any issues with Java or maven versions.

Limitations:

* There is no count of word repetition
* Named entities are exact match based on input
* Application currently doesn’t recognize position of sentence or word class. E.g. DT,NN, etc
* There is no lemmatization
* There is no punctuation removal or detection for words
* Currently very limited.

References:

[http://www3.cs.stonybrook.edu/~ychoi/cse628/lecture/05-pos.pdf](http://www3.cs.stonybrook.edu/~ychoi/cse628/lecture/05-pos.pdf" \t "_blank)

[http://nlp.stanford.edu](http://nlp.stanford.edu/research.shtml" \t "_blank)

Application Architecture summary:

Normally I start by writing unit tests but due to my inexperience in the domain, I jumped into problem solving so unfortunately I didn’t write any tests.

Processing:

My original plan was to create a little demo video stepping through the code and explaining it but I had limited time this weekend. I’ll be happy to do the video and send it or could do it over webex if needed.

App.groovy starts the application by calling the XMLGenerator service (or the ConcurrencyService incase of multi-threading). The XMLGenerator service calls the InputReader service to read the input data then the paragraph service to divide it into paragraph then the sentence service to divide it into sentences and so on…

The application uses regular expressions to tokenize the input data.

Code structure:

 App.groovy - main App class to run the program

*config* - contains the configuration objects.

* ConfigManager: A singleton for accessing java properties files and allow for externalizing configurations such as location of input files

*domain* - contains the domain objects. It’s our model for what we care about in order to process human language.

* Paragraph: biggest block to be identified/tokenized and is defined by the occurrence of a new line
* Sentence: second biggest block, rules for detection are defined in a member containing the regular expression used. It has a reference to its containing paragraph
* Word: smallest unit or token in the current design, also has a regex to define the rules to recognition. Has a reference to the containing sentence and containing paragraph
* NamedEntity: consists of name and a reference to its containing sentence

eums

* NLPConstants: contains the characters used in the regex to try to make the code more readable.

logging - contains Logback error logger

* ErrorLogger

service - contains the services that do all the important work for the application. Defined as singletons.

* ParagraphService: takes in all input and splits it into paragraphs
* SentenceService: takes in paragraphs and splits them into sentences, returns an ArrayList of the sentence domain object
* WordService: takes in sentences and splits them into words, returns an ArrayList of the word domain object
* NERService: loads named entities from an input file and also finds named entities in all input sentences
* XMLGenerator: calls the services injected in order to produce the xml output
* ConcurrencyService: multi-threads any statement by entering the number of desired threads.
* InputReader: Reads input files and directories